(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

(12) **Offenlegungsschrift**

(10) **DE 101 46 356 A 1**

(51) Int. Cl.[7]:
H 03 M 7/30
G 06 T 9/00

(21) Aktenzeichen: 101 46 356.1
(22) Anmeldetag: 20. 9. 2001
(43) Offenlegungstag: 24. 4. 2003

DE 101 46 356 A 1

(71) Anmelder:
Syntion AG, 80636 München, DE

(74) Vertreter:
Meissner, Bolte & Partner, 81679 München

(72) Erfinder:
Matzner, Rolf, Dr., 80995 München, DE;
Gemeinhardt, Lars, 09116 Chemnitz, DE

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) Verfahren zum Komprimiern von dynamischen Webseiten und eine Datenverarbeitungseinrichtung zur Durchführung des Verfahrens

(57) Zum Komprimieren von dynamischen Webseiten wird vorgeschlagen, die dynamischen Webseiten auf statische Blöcke zu untersuchen und nur die statischen Blöcke, nicht aber in jedem Fall auch die dynamischen Blöcke zu komprimieren, es sei denn, die Belastung der CPU des Webservers erlaubt dies. Die statischen Blöcke werden in komprimierter Form abgespeichert, so daß beim erneuten Abrufen keine erneute Kompression, sondern lediglich ein Ersetzen der unkomprimierten durch die komprimierten statischen Blöcke erfolgen muß.

DE 101 46 356 A 1

Beschreibung

[0001] Die Erfindung betrifft ein Verfahren zum Komprimieren von dynamischen Webseiten sowie eine Datenverarbeitungseinrichtung zur Durchführung des Verfahrens.

[0002] Dynamische Webseiten bestehen aus Inhalten, die zum Teil ständig aktualisiert bzw. auf einen Nutzer angepaßt werden. Eine ständige Aktualisierung erfolgt beispielsweise bei Webseiten, welche Börsenkurse wiedergeben. Eine auf den Nutzer angepaßte Änderung erfolgt beispielsweise bei Webseiten im E-Commerce-Bereich.

[0003] Derartige dynamische Webseiten können als Template also als Schablonen oder lediglich als Beschreibung von Aufbau und Inhalt der Webseite auf dem Webserver abgelegt sein.

[0004] Ist eine Webseite als Template abgelegt, ist dessen Struktur vorgegeben. Eine solche Webseite kann daher entsprechend ihres Inhalts in statische und dynamische Blöcke unterteilt werden. Der Inhalt der dynamischen Blöcke ist durch eine Abfolge von Befehlen in einer Programmier-, eine Skript- oder auch einer Kommandosprache beschrieben.

[0005] Bei einem Abruf der Seite durch einen Nutzer ersetzt der Application-Server die Beschreibung durch die aktuellen Inhalte. Aussehen und Aufbau bleiben bei einer aus einem Template generierten Webseite unverändert, wie dies in **Fig.** 1 gezeigt ist.

[0006] Statische Blöcke enthalten Inhalte, die nie oder in sehr großen Zeitabständen generiert werden. Diese Blöcke sind oft in HTML geschrieben.

[0007] Die Inhalte dynamischer Blöcke werden in kurzen Zeitabständen, z. B. bei jedem Abruf der Webseite, generiert. Die Inhalte sind zunächst nur beschrieben. Erst beim Generieren wird diese Beschreibung dann durch Daten ersetzt.

[0008] Die Beschreibung kann durch eine Programmier-, eine Skript- oder auch eine Kommandosprache erfolgen. Gewöhnlich werden Skriptsprachen verwendet, da diese sich gut in HTML integrieren lassen. Beispiele für Skriptsprachen sind:

- PHP (abgeleitet aus Hypertext Preprocessor) als eine offene serverseitige Skriptsprache, die jeder frei nutzen und ändern kann oder
- JSP (Java Server Pages)

[0009] Der Nutzer ruft mit dem Eingeben und Bestätigen einer URL-Adresse im Browser eine Webseite auf. Das bedeutet, der Browser sendet an den Webserver ein Request (Anfrage). Dieser Request enthält neben der URL der gewünschten Webseite eine Fülle von Informationen über die Fähigkeiten des Browsers, z. B. über die unterstützten Kompressionsverfahren, MIME-Typen oder den Browser Typ. Der Webserver wertet den Request aus und versucht, die zu übertragenen Daten an den Browser anzupassen.

[0010] Statische Webseiten sind in der Regel auf dem Webserver in komprimierter Form (GZip- und Compress-Format) abgelegt. GZip und Compress sind die gängigen Kompressionsverfahren für statische Webseiten. Entsprechend den Fähigkeiten des Browsers wird die Webseite im GZip- oder Compress-Format übertragen.

[0011] Unterstützt der Browser kein Kompressionsverfahren, überträgt der Webserver die Webseite unkomprimiert.

[0012] Die Anfrage zu einer dynamischen Webseite reicht der Webserver zum Application-Server weiter. Dieser übernimmt das Generieren der dynamische Blöcke. Das heißt, er ersetzt die beschriebenen dynamischen Blöcke durch deren Inhalt. Die statischen Blöcke der Webseite werden unkomprimiert eingefügt. Der Application-Server übergibt die vollständige dynamische Webseite dem Webserver. Dort kann eine Komprimierung erfolgen. Am Ende sendet der Webserver die Webseite zum Browser, wie dies **Fig.** 2 zeigt.

[0013] Der Umfang und die Bedeutung dynamischer Webseiten im Internet nimmt aufgrund der vielfältigen Einsatzmöglichkeiten ständig zu. Beispiele sind dynamische Webseiten, die aktuelle Aktienwerte bereitstellen oder bei denen Wetterinformationen abgerufen werden können.

[0014] Dynamische Webseiten stellen aber an die Webserver bedingt durch ihre Eigenschaften größere Anforderungen. Die Schwierigkeit besteht im Bereitstellen komprimierter dynamischer Webseiten unter Berücksichtigung der Effizienz des Webservers.

[0015] Zwei wichtige Größen für die Effizienz eines Webservers sind die Verkehrslast und die CPU-Belastung. Ziel ist, daß beide Größen in einem ausgewogenen Verhältnis zueinander stehen und das stets eine Reserve für auftretende Spitzen verfügbar ist.

[0016] Das Problem bei dynamischen Webseiten ist, daß sie sich nicht wie statische Webseiten vorkomprimieren und auf dem Webserver ablegen lassen. Damit können sie auf die zwei Größen entscheidenden Einfluß haben. Folgende Situationen sind möglich:

Die dynamischen Webseiten werden nicht komprimiert und erhöhen damit aufgrund des größeren Datenumfangs massiv die Verkehrslast des Webservers.

[0017] Die dynamischen Webseiten werden erst unmittelbar nach deren Generierung mit einem bestehendem Verfahren komprimiert. Der Nachteil ist eine hohe CPU-Belastung des Webservers.

[0018] Das Diagramm gemäß **Fig.** 3 verdeutlicht die Problematik:

Der Erfindung liegt die Aufgabe zugrunde, ein Verfahren zum Komprimieren von dynamischen Webseiten sowie eine Datenverarbeitungsvorrichtung für diesen Zweck vorzuschlagen, mittels derer in einfacher Weise eine Kompression bei verbesserter Ausnutzung des Ressourcenverbrauchs sichergestellt wird.

[0019] Diese Aufgabe wird durch ein Verfahren nach Anspruch 1 und eine Datenverarbeitungseinrichtung nach Anspruch 21 gelöst.

[0020] In der nachfolgenden Beschreibung wird das erfindungsgemäße Verfahren als "SSC" (Server Side Compression) bezeichnet.

[0021] Ein wesentlicher Punkt der Erfindung liegt darin, daß nicht "blind" komprimiert wird, sondern daß nur diejenigen Daten komprimiert werden, welche einen wesentlichen Anteil der gesamten zu übertragenden Datenmenge bilden und die andererseits mit verringertem Aufwand komprimierbar sind. Die statischen Daten werden nämlich nur bei einem

"Neuaufbau" einer Webseite geändert, bleiben also über lange Zeit unverändert erhalten. Die können darum in komprimierter Form abgespeichert und immer wieder verwendet werden.

[0022] Die Erfindung wird nachfolgend unter Bezugnahme auf die Abbidungen näher beschrieben. Hierbei zeigen:

[0023] **Fig.** 1 einen Seitenabrufvorgang;

[0024] **Fig.** 2 einen herkömmlichen Aufbau der Hardware;

[0025] **Fig.** 3 ein Diagramm zur Hardwarebelastung bei herkömmlichen Techniken;

[0026] **Fig.** 4 ein Diagramm zur Hardwarebelastung beim erfindungsgemäßen Verfahren;

[0027] **Fig.** 5 einen herkömmlichen Datenfluß;

[0028] **Fig.** 6 einen Datenfluß beim erfindungsgemäßen Verfahren;

[0029] **Fig.** 7 einzelne Schritte beim Entstehen eines SSC-Dokuments;

[0030] **Fig.** 8 einen Tag-Einfügevorgang;

[0031] **Fig.** 9 den Aufbau einer Block-Datei;

[0032] **Fig.** 10 einen 1. Kompressionsvorgang;

[0033] **Fig.** 11 einen 2. Kompressionsvorgang;

[0034] **Fig.** 12 ein SSC-Dokument und

[0035] **Fig.** 13 einen schematischen Aufbau einer Datenverarbeitungseinrichtung gemäß der Erfindung.

[0036] Eine Analyse dynamischer Webseiten in bezug auf die Zusammensetzung der Daten erbrachte folgendes Ergebnis: Der Anteil dynamischer Daten im Verhältnis zu den statischen ist wesentlich geringer. Werden von einer dynamischen Webseite also lediglich die statischen Daten komprimiert und die dynamischen unkomprimiert übertragen, ist keine wesentliche Verringerung des Komprimierungsfaktors gegenüber einer Komprimierung der kompletten Webseite zu erwarten.

[0037] Diesen Effekt nutzt SSC. Bei dem Verfahren wird auf eine bessere Komprimierung zugunsten einer geringeren CPU-Belastung verzichtet. Es ist ein Kompromiss zwischen den beiden oben beschriebenen Möglichkeiten. Das Ergebnis veranschaulicht das Diagramm gemäß **Fig.** 4:
Das Verhältnis von Verkehrslast und CPU-Belastung ist nahezu ausgeglichen, der Ressourcenverbrauch des Webservers ist gering.

[0038] Insbesondere wird also mit der Erfindung ein Verfahren zum Komprimieren von dynamischen Webseiten aufgezeigt, insbesondere zum Komprimieren von Webseiten, die mindestens einen statischen Block und mindestens einen dynamischen, insbesondere durch Abruf von einem Nutzer veränderlichen Block aufweisen, die auf einem Webserver und einem Application-Server vorgehalten oder generiert werden. Beim erfindungsgemäßen Verfahren werden nun folgende Schritte vorgenommen:
Es wird ein Datensatz aufgenommen, der mindestens einen statischen und einen dynamischen Block umfaßt.

[0039] Es wird jeweils eine Identität aller im Datensatz vorhandenen statischen Blöcke, also ein "Fingerprint" festgestellt.

[0040] Es wird festgestellt, ob statische Blöcke mit derselben Identität (mit demselben Fingerprint) in einem Blockspeicher schon als komprimierte Blöcke gespeichert sind. Wenn dies der Fall ist, so wird der im Datensatz vorhandene statische Block durch den entsprechenden komprimierten Block ersetzt. Wenn dies nicht der Fall ist, so wird der statische Block komprimiert und im Blockspeicher als komprimierter Block abgelegt. Der komprimierte Block ersetzt dann wieder den im Datensatz vorhandenen statischen Block.

[0041] Abschließend wird ein Sendedatensatz abgesendet, bei welchem die statischen Blöcke durch komprimierte Blöcke ersetzt sind, wobei der Sendedatensatz selbstverständlich auch die dynamischen Blöcke enthält.

[0042] Vorzugsweise wird die Identität eines statischen Blocks anhand einer, aus ihm selbst gewonnenen Kennung festgestellt. Diese Kennung soll so sein, daß eine Verwechslung mit anderen statischen Blöcken (Blöcken anderen Inhalts) unwahrscheinlich ist. Die aus der Identität gewonnene Kennung wird jedem komprimierten Block zugehörig im Blockspeicher gespeichert und beim Feststellen der Identität zum Vergleich mit den Identitäten der statischen Blöcke im Datensatz verwendet. Es werden also nicht die statischen Blöcke selbst miteinander verglichen sondern – was eine wesentliche Verringerung des Aufwandes bedeutet – lediglich ihre Kennungen. Es kann als Kennung eine Prüfsumme über den statischen Block, insbesondere eine CRC-32-Summe verwendet werden.

[0043] CRC-32 (Cyclic Redundancy Check mit 32-Bit-Prüfsumme) ist ein gebräuchliches mathematisches Verfahren zum Ermitteln einer Prüfsumme. Mit Hilfe dieser Prüfsumme können Identitätsprüfungen durchgeführt oder wie bei SSC Datenmengen verglichen werden. Der Vorteil dieses Verfahrens ist, daß eine geringe Abweichung bei den Daten eine große Änderung der Prüfsumme bewirkt und damit auch kleinste Fehler nicht übersehen werden können.

[0044] Die Verkehrslast bestimmt die Datenmenge, die innerhalb eines Zeitintervalls vom Webserver abgefordert wird. Sie ist abhängig von der Anzahl der Requests (Anfragen zu Webseiten, die auf dem Webserver abgelegt sind) und von der Datenmenge, die als Antwort pro Request zum Client übertragen wird. Während eine hohe Anzahl an Requests erwünscht ist, versucht man die zu übertragende Datenmenge durch Kompressionsverfahren möglichst gering zu halten, um damit die Ressourcen des Webserver nicht unnötig zu blockieren.

[0045] Vorzugsweise umfaßt die Kennung die Länge des statischen Blocks.

[0046] Zusätzlich zu jedem komprimierten Block wird vorzugsweise der dazu gehörige statische Block abgespeichert. Bei Bedarf steht dieser somit zur Verfügung.

[0047] Die Daten werden vorzugsweise nach dem (an sich bekannten) DEFLATE-Verfahren komprimiert. Durch die Verwendung dieses bekannten Verfahrens ist eine Dekompression bei einem Empfänger bzw. Nutzer leicht möglich.

[0048] Die dynamischen Blöcke kann man im wesentlichen unverändert, insbesondere unkomprimiert im Datensatz senden. Wenn nun aber der Server zum Zeitpunkt, zu welchem die Webseite abgerufen wird, nur wenig ausgelastet ist, so kann man die dynamischen Blöcke (oder einige der dynamischen Blöcke) ebenfalls komprimieren und als komprimierte dynamische Blöcke in den Sendedatensatz einfügen. Es erfolgt also eine Komprimierung "on fly". Dadurch werden die vorhandenen Ressourcen optimal genutzt, eine Überlastung des Servers jedoch vermieden.

[0049] Weiterhin besteht die Möglichkeit, dynamische Blöcke nicht nur unkomprimiert oder "on the fly" komprimiert

(je nach Serverlast) auszuliefern, sondern dynamische Blöcke, welche durch ein bestimmtes Merkmal (z. B. ihrer Größe) herausstechen, vorkomprimiert (wie die statischen Blöcke) abzulegen. Dies wird im Fall des Merkmales der Größe bei Überschreiten eines Schwellwerts erfolgen.

[0050] Weiterhin ist sinnvoll, für jede Kompression eines Blockes eine erforderliche Mindestgröße (untere Schranke) zu definieren, welche überschritten werden muß, damit eine Kompression überhaupt durchgeführt wird. So können z. B. beim Server einer Zeitung Artikel aus einer Datenbank heraus generiert werden, die also dynamisch sind. Andererseits ändert sich ein Artikel im Regelfall nicht mehr. Es macht also in diesem Fall Sinn, ihn hinsichtlich Kompression wie einen statischen Block zu behandeln.

[0051] Jedem letzten Block eines Datensatzes wird ein Endblock zur Bezeichnung des Endes des Datensatzes hinzugefügt, was die Handhabbarkeit der Datensätze erleichtert.

[0052] Jedem Block des Datensatzes wird ein Tag zur Kennzeichnung als dynamischer oder als statischer Block hinzugefügt, wobei diese Hinzufügung vorzugsweise an erster Stelle des Blocks erfolgt. Die Tags werden vorzugsweise als HTML-Kommentar ausgebildet.

[0053] Vorzugsweise werden die Tags von einem Parser eingefügt, der auf dem Application-Server läuft und auf abgelegte Templates zugreift, die er Block für Block analysiert und denen er die Tags hinzufügt. Die mit den Tags markierte noch unkomprimierte Webseite wird einem Pre-Compressor zugeführt, der entsprechend markierte Blöcke komprimiert, die zusammen mit unkomprimierten Blöcken dann einem Post-Compressor übergeben werden. Der Post-Compressor erstellt schritthaltend zur Datenbearbeitung durch den Pre-Compressor, also verzahnt mit diesem arbeitend ein neues Dokument und zwar vorzugsweise nach dem GZip-Format. Nach Bildung eines GZip-Headers reiht der Post-Compressor die vom Pre-Compressor übergebenen Blöcke unmittelbar mit der Übergabe in das neue Dokument ein, was eine hocheffektive und schnelle Bearbeitung ermöglicht.

[0054] Der Post-Compressor kennzeichnet unkomprimierte Blöcke durch einen Header und zwar vorzugsweise einen solchen nach dem DEFLATE-Format.

[0055] Der Post-Compressor erstellt einen zusätzlichen END-Block und fügt diesen zur Kennzeichnung eines Endes der Datenblöcke dem neuen Dokument hinzu. Weiterhin errechnet der Post-Compressor eine Checksumme, insbesondere eine CR-32-Prüfsumme und die Größe des unkomprimierten neuen Dokuments für einen Trailer, der ebenfalls hinzugefügt bzw. gesendet wird.

[0056] Die erfindungsgemäße Datenverarbeitungseinrichtung zum Komprimieren von dynamischen Webseiten, insbesondere zum Komprimieren von dynamischen Webseiten, die mindestens einen statischen Block und mindestens einen, insbesondere durch Abruf von einem Nutzer veränderlichen dynamischen Block aufweisen, umfaßt einen Webserver und einen Application-Server. Es ist ein Precompressor vorgesehen, der so ausgebildet ist, daß bei Aufforderung durch den Webserver an den Application-Server, eine dynamische Webseite zu generieren, der Precompressor eine noch unkomprimierte Website in folgenden Schritten bearbeitet:

Es wird festgestellt, ob statische Blöcke mit derselben Identität (mit demselben Fingerprint) in einem Blockspeicher schon als komprimierte Blöcke gespeichert sind. Wenn dies der Fall ist, so wird der im Datensatz vorhandene statische Block durch den entsprechenden komprimierten Block ersetzt. Wenn dies nicht der Fall ist, so wird der statische Block komprimiert und im Blockspeicher als komprimierter Block abgelegt. Der komprimierte Block ersetzt dann wieder den im Datensatz vorhandenen statischen Block.

[0057] Abschließend wird ein Sendedatensatz abgesendet, bei welchem die statischen Blöcke durch komprimierte Blöcke ersetzt sind, wobei der Sendedatensatz selbstverständlich auch die dynamischen Blöcke enthält.

[0058] Vorzugsweise wird die Identität eines statischen Blocks anhand einer, aus ihm selbst gewonnenen Kennung festgestellt. Diese Kennung soll so sein, daß eine Verwechslung mit anderen statischen Blöcken (Blöcken anderen Inhalts) unwahrscheinlich ist. Die aus der Identität gewonnene Kennung wird jedem komprimierten Block zugehörig im Blockspeicher gespeichert und beim Feststellen der Identität zum Vergleich mit den Identitäten der statischen Blöcke im Datensatz verwendet. Es werden also nicht die statischen Blöcke selbst miteinander verglichen sondern – was eine wesentliche Verringerung des Aufwandes bedeutet – lediglich ihre Kennungen. Vorzugsweise wird als Kennung eine Prüfsumme über den statischen Block, insbesondere eine CRC-32-Summe verwendet.

[0059] Für SSC werden drei SSC-Komponenten in die bestehende Architektur auf der Serverseite eingefügt – Pre-Processor, Pre-Compressor and Post-Compressor. Der Pre-Processor ist fest mit dem Application-Server verbunden, Pre- und Post-Compressor sind vom Application-Server unabhängig.

[0060] Die drei SSC-Komponenten klinken sich in den Prozess der Erstellung bzw. der Übertragung einer dynamischen Webseite ein. Sie werten jeden Block einer bereits generierten Seite aus, prüfen, ob der Block statisch oder dynamisch ist bzw. die Daten komprimiert oder unkomprimiert übertragen werden sollen, und formatieren die Daten am Ende nach den GZip-Konventionen.

[0061] Pre-Processor, Pre-Compressor und Post-Compressor sind die SSC-Komponenten, die in die serverseitige Architektur eingefügt werden. Die Anordnung der SSC-Komponenten lässt sich durch den logischen Datenfluß vom Generieren einer dynamischen Webseite verdeutlichen.

[0062] Die Grafik gemäß **Fig.** 5 zeigt den logischen Datenfluß ohne die SSC-Komponenten.

[0063] Der Webserver fordert den Application-Server auf, eine dynamische Webseite zu generieren. Der Parser analysiert das Template der dynamischen Webseite auf seine statischen und dynamischen Blöcke. Der Application-Server fügt dann die Inhalte der Blöcke ein und übergibt anschließend die fertige dynamische Webseite dem Webserver.

[0064] Die Grafik gemäß **Fig.** 6 verdeutlicht den logischen Datenfluß mit SSC-Komponenten:

Der Pre-Processor schließt sich unmittelbar dem Parser an bzw. verschmilzt mit diesem. Pre- und Post-Compressor – ebenfalls eng verknüpft – bearbeiten die Daten nach dem Generieren durch den Application-Server und übergeben das SSC-Dokument dem Webserver. Pre- und Post-Compressor können – müssen aber nicht auf dem Application-Server laufen.

[0065] Den allgemeinen Prozeßablauf beschreibt der folgende Abschnitt.

[0066] Der Webserver wertet den Request eines Nutzers aus. Hat dieser eine dynamische Webseite angefordert, leitet

er die Anfrage an den Application-Server weiter. Der Parser des Application-Server greift auf das abgelegte Template dieser Seite zu und analysiert es auf seine statischen und dynamischen Blöcke. Gleichzeitig fügt der Pre-Processor an den Anfang von jedem Block ein entsprechendes SSC-Tag ein. Dieses SSC-Tag ist eine SSC-spezifische Markierung, die den Block für SSC eindeutig als statisch bzw. dynamisch kennzeichnet.

[0067] Danach generiert der Application-Server die Inhalte der Blöcke, ohne die eingefügten SSC-Tags zu entfernen. Die fertige aber unkomprimierte dynamische Webseite übergibt er dem Pre-Compressor. Dieser wertet die Webseite anhand der SSC-Tags Block für Block aus. Das bedeutet:

- Er weist jedem Block eine Block-ID zu. Diese setzt sich aus der Länge des Blocks und einer errechneten Checksumme zusammen. Diese ermöglicht die eindeutige Erkennung des Blocks.
- Soll der Block komprimiert werden, prüft der Pre-Compressor anhand der Block-ID, ob es für diesen Block bereits eine Block-Datei gibt. Die Block-Datei enthält den Inhalt eines Blocks komprimiert und unkomprimiert. Findet der Pre-Compressor die Block-Datei, ersetzt der Pre-Compressor den unkomprimierten Block der Webseite durch den komprimierten Block aus der Block-Datei. Wenn nicht, komprimiert der Pre-Compressor den Block nach dem DEFLATE-Verfahren. Zusätzlich erstellt er eine Block-Datei und legt diese für weitere Verwendungen ab.

[0068] Parallel zum Pre-Compressor erstellt der Post-Compressor ein SSC-Dokument nach dem GZip Format, in das er die vom Pre-Compressor blockweise übergebenen Inhalte einfügt. Das fertige SSC-Dokument wird über den Webserver zum Browser des Nutzers übertragen.

[0069] **Fig. 7** stellt die einzelnen Schritte beim Entstehen eines SSC-Dokuments aus dem Template der dynamischen Webseite dar.

[0070] Jeder Block der dynamischen Webseite wird dem DEFLATE-Verfahren unterzogen. D. h., es komprimiert die Blöcke zunächst nach festem und nach dynamischem Huffmann-Code und vergleicht die Ergebnisse mit dem Umfang der unkomprimierten Daten. Das Ergebnis mit dem besten Komprimierungsfaktor wird verwendet. Wird durch das Komprimieren keine Verringerung des Datenumfangs erreicht, verarbeitet das DEFLATE-Verfahren die Daten unkomprimiert.

[0071] Das DEFLATE-Verfahren fügt jedem Block einen Header hinzu, deren Aufbau sich je nach verwendeten Daten unterscheidet. Die Header sind nachfolgend beschrieben. Der Endblock ist eine Besonderheit von SSC zum Kennzeichnen des letzten Blocks.

[0072] Das DEFLATE-Verfahren ist eine Kombination des LZ77 (Lempel-Ziv) Algorithmus und dem Kodieren nach Huffmann. Es ist in der Spezifikation RFC 1951 "DEFLATE Compressed Data Format Specification version 1.3" von Peter Deutsch beschrieben und definiert.

[0073] Von dem Header eines komprimierten Blocks sind die ersten drei Bits des ersten Byte wie folgt definiert:

					BTYPE		BFINAL
Bit	7	6	5	4	3	2	1	0

[0074] Die Bits bedeuten im einzelnen:

Bit	SSC-Wert	Beschreibung
0 (BFINAL)	0	Flag, das nur beim letzten Block des SSC-Dokuments gesetzt wird.
1-2 (BTYPE)	00 01 oder 10	Spezifiziert die nachfolgenden Daten: Unkomprimierte Daten Komprimierte Daten mit festen Huffmann-Code Komprimierte Daten mit dynamischen Huffmann-Code
3-7		Ersten Bits der komprimierten Daten

[0075] Bei unkomprimierten Datenblöcken sind die ersten fünf Bytes definiert. Die ersten drei Bits von Byte **1** kennzeichnen einen Block als unkomprimiert. Daten bleiben unkomprimiert, wenn dies entsprechend konfiguriert wurde und/oder wenn die Ergebnisse der zwei möglichen Kompressionsverfahren (fester und dynamischer Huffmann-Code) zu keiner Reduzierung des Datenumfangs führt. Der Header setzt sich folgendermaßen zusammen:

	Byte 1							2	3	4	5
0	0	0	0	0	BTYPE		BFINAL	LEN		NLEN	
Bit 7	6	5	4	3	2	1	0				

[0076] Die Bits von Byte **1** bedeuten im einzelnen:

Bit	SSC-Wert	Beschreibung
0 (BFINAL)	0	Flag, das nur beim letzten Block des SSC-Dokuments gesetzt wird.
1-2 (BTYPE)	00	Bitfolge für unkomprimierte Daten
3-7	0	Bits zum Auffüllen des 1. Bytes

[0077] Bytes **2** und **3** beschreiben den Wert für die Länge des unkomprimierten Blocks. Bytes **4** und **5** beschreiben den Wert für die Länge des unkomprimierten Blocks minus 1.

[0078] Für den letzten Block innerhalb eines SSC-Dokuments muß nach der DEFLATE-Spezifikation das erste Bit (der BFINAL-Flag) auf 1 gesetzt sein.

[0079] Während der blockweisen Bearbeitung der Daten ist es für SSC nicht möglich, automatisch den letzten Block zu erkennen. Daher ist das BFINAL-Flag der Datenblöcke immer 0. SSC fügt am Ende einen zusätzlichen Block, den Endblock, ein und setzt das BFINAL-Flag auf 1. Dieser Endblock enthält keine Daten.

[0080] Der Pre-Processor ist eng mit dem Parser auf dem Application-Server verknüpft und muß daher auch auf diesem laufen.

[0081] Er markiert als ersten Schritt von SSC jeden Block einer dynamischen Webseite mit einem SSC-spezifischen Tag. Diese Tags sind Voraussetzung für den weiteren SSC-Prozess. Der Pre-Processor kann nur templatebasierte dynamischen Webseiten bearbeiten.

[0082] Die SSC-spezifischen Tags dienen dem Erkennen der Blöcken einer dynamischen Webseite und dem Kennzeichnen dieser als statisch oder dynamisch. Der Pre-Processor fügt sie an erste Stelle eines Blocks ein.

[0083] Die zwei SSC-spezifischen Tags (statisch und dynamisch) entsprechen von ihrer Syntax her einem HTML Kommentar. Daraus ergibt sich der Vorteil, daß diese Tags als solche vom Application-Server erkannt aber nicht dargestellt werden. Beispiele für SSC-spezifische Tags sind:

- statisch: <!--/@-->
- dynamisch: <!--\@/-->

[0084] Mit dem Abruf einer templatebasierten dynamischen Webseite greift der Parser des Application-Server auf das abgelegte Template zu. Der Parser analysiert das Template Block für Block auf deren Charakter (statisch oder dynamisch). Gleichzeitig fügt der Pre-Processor das entsprechende statische oder dynamische SSC-spezifische Tag an den Beginn des Blocks ein. Durch diesen parallelen Ablauf bleibt die benötigte zusätzliche Rechenleistung gering.

[0085] **Fig. 8** zeigt beispielhaft das Einfügen der SSC-spezifischen Tags. <? ist das PHP-Anfangstag und ?> das Endtag für den dynamischen Block.

[0086] Das so analysierte und markierte Template wird anschließend durch den Application-Server generiert. Die danach vollständige unkomprimierte dynamische Webseite mit den eingefügten SSC-spezifischen Tags übernimmt der Pre-Compressor.

[0087] Der Pre-Compressor setzt eine mit SSC-spezifischen Tags vorbereitete dynamische Webseite voraus. Er übernimmt diese mit generiertem Inhalt vom Application-Server.

[0088] Er überprüft für jeden Block dieser Webseite:

- den Blocktyp (anhand des SSC-spezifischen Tag),
- ob der Block komprimiert werden soll bzw. ob für diesen Block bereits eine Block-Datei erstellt und abgelegt wurde.

[0089] Zu komprimierende Blöcke bearbeitet der Pre-Compressor nach dem DEFLATE-Verfahren und übergibt sie dem Post-Compressor. Blöcke, die unkomprimiert bleiben, übergibt er sofort dem Post-Compressor.

[0090] Der Pre-Compressor ist zusammen mit dem Post-Compressor unabhängig vom Application-Server. D. h., beide können auch auf anderen Servereinheiten, z. B. dem Webserver, integriert sein.

[0091] Der Pre-Compressor weist jedem zu komprimierendem Block eine eindeutige Block-ID zu. Diese Block-ID besteht aus zwei Bestandteilen: der CRC-32 Prüfsumme und der Länge des unkomprimierten Blocks:
Block-ID = <CRC-32><Länge des Blocks>

[0092] Das Kreuzprodukt der CRC-32 Prüfsumme und der Länge des unkomprimierten Blocks ergibt eine Block-ID, mit der sich ein Block eindeutig bestimmen läßt und damit Verwechslungen mit hoher Wahrscheinlichkeit ausschließt.

[0093] Für jeden Block einer dynamischen Webseite, der komprimiert werden soll, erstellt der Pre-Compressor eine Block-Datei. Diese Block-Datei ist in einem bestimmten Verzeichnis abgelegt (gecached) und besteht aus dem Block in unkomprimierter und in nach dem DEFLATE-Verfahren komprimierter Form. Der Name der Block-Datei wird von der Block-ID geprägt, die Endung kann dem Charakter des Blocks entsprechen. Beispiel für eine Kennung ist: SSC-cache-<Länge>-<CRC-32>.static

[0094] **Fig. 9** zeigt den Aufbau einer Block-Datei.

[0095] Der Pre-Compressor bearbeitet die vom Application-Server übernommene dynamische Webseite blockweise. Dabei prüft er als erstes, ob der Block komprimiert abgelegt werden soll oder nicht. Wenn nicht übernimmt der Post-Compressor den unkomprimierten Inhalt des Blocks. Ist eine Komprimierung gefordert, errechnet der Pre-Compressor für den Block die Block-ID und prüft mit dieser Block-ID, ob für diesen Block bereits eine Block-Datei erstellt und ab-

gelegt ist. Entsprechend dem Ergebnis ergeben sich 2 Fallsituationen.
Fall 1 (**Fig.** 10): Der Pre-Compressor findet die entsprechende Block-Datei. Der Post-Compressor übernimmt den komprimierten Block aus dieser Block-Datei.
Fall 2 (**Fig.** 11): Es gibt noch keine entsprechende Block-Datei. Der Pre-Compressor komprimiert den Block nach dem DEFLATE-Verfahren und übergibt diesen den Post-Compressor. Zusätzlich erstellt er eine Block-Datei und legt diese ab.

[0096] Der Post-Compressor übernimmt vom Pre-Compressor die komprimierten bzw. unkomprimierten Blöcke und erstellt aus diesen ein SSC-Dokument.

[0097] Der Post-Compressor ist zusammen mit dem Pre-Compressor unabhängig vom Application-Server. D. h., beide können auch in anderen Servereinheiten, z. B. dem Webserver, integriert sein.

[0098] Das SSC-Dokument setzt sich, wie aus **Fig.** 12 erkennbar, aus einem Header, den einzelnen Datenblöcken und einem Trailer zusammen. Dieser Aufbau entspricht dem GZip-Format, der in der Spezifikation RFC1952 "GZIP file format specification version 4.3" von Peter Deutsch definiert und beschrieben ist.

[0099] Der Aufbau des Header (Kopf) vom SSC-Dokument setzt sich folgendermaßen zusammen:

Byte

1	2	3	4	5	6	7	8	9	10
ID1	ID2	CM	FLG	MTIME				XFL	OS

[0100] Die einzelnen Bytes bedeuten:

Byte	SSC-Wert	Beschreibung
1	$1F_{HEX}$	ID1 - erstes Byte zur Identifizierung des gzib-Formats
2	$8B_{HEX}$	ID2 - zweites Byte zur Identifizierung des gzib-Formats
3	8	CM - Byte bestimmt Komprimierungsmethode (deflate)
4	0	FLG – Einstellungen
5 - 8	0	MIME - Zeit der Komprimierung der Datei für SSC alle 4 Bytes 0 → kein Zeitstempel
9	0	XFL - Flag für spezifische Komprimierungsmethoden
10	0	OS - Byte zur Identifizierung des Betriebssytems

[0101] Das SSC-Dokument endet mit einem 8 Byte großem Trailer (Anhang). Diese Bytes haben folgende Bedeutung:

Byte	Beschreibung
1 - 4	CRC32 – Checksumme
5 - 8	ISIZE - Größe des unkomprimierten Datenblocks

[0102] Der Post-Compressor erstellt parallel zur Auswertung der dynamischen Webseite durch den Pre-Compressor ein neues Dokument nach dem GZip-Format. Dabei bildet der Post-Compressor zunächst den GZip-Header. Anschließend reiht er unmittelbar mit der Übergabe der Blöcke durch den Pre-Compressor diese Block für Block in das Dokument ein.

[0103] Unkomprimierte Blöcke, die der Pre-Compressor ohne Bearbeitung weiterleitet, kennzeichnet der Post-Compressor durch einen Header entsprechend dem DEFLATE-Format.

[0104] Sind alle Blöcke der dynamischen Webseite abgearbeitet, ergänzt der Post-Compressor einen zusätzlichen End-Block, der keine Daten enthält. Das BFINAL-Flag (BFINAL = 1) im Header des End-Blocks kennzeichnet das Ende der Datenblöcke.

[0105] Als letzten Schritt errechnet der Post-Compressor die Checksumme und die Größe des unkomprimierten Datenblocks für den Trailer.

[0106] Das fertige SSC-Dokument übergibt der Post-Compressor dem Webserver, der dieses zum Nutzer weiterleitet.

[0107] In der beiliegenden **Fig.** 13 ist ein Ausführungsbeispiel der erfindungsgemäßen Datenverarbeitungseinrichtung im Prinzip gezeigt. Hierbei ist ein Webserver **10** mit einem Application-Server **11** verbunden, auf welchem ein Parser **12** läuft. Der Parser **12** ist eng mit einem Pre-Processor verbunden, läuft also ebenfalls auf dem Application-Server **11**. Der Pre-Processor (mit dem Parser) bearbeitet die dynamischen, templatebasierten Webseiten.

[0108] Es ist ein Pre-Compressor **14** vorgesehen, der zusammen mit einem Post-Compressor **15** vorzugsweise unabhängig vom Application-Server ist, also mit dem Pre-Compressor **14** z. B. im Webserver integriert sein kann. Schließlich ist ein Blockspeicher **13** vorgesehen, der mit dem Pre-Compressor **14** verbunden ist, so daß der Pre-Compressor **14** den Blockspeicher **13** auf dort schon gespeicherte komprimierte statische Blöcke untersuchen bzw. statische Blöcke nach dem Komprimieren dort ablegen kann.

Bezugszeichenliste

10 Webserver
11 Application-Server
12 Parser
13 Blockspeicher

14 Pre-Compressor
15 Post-Compressor

Patentansprüche

1. Verfahren zum Komprimieren von dynamischen Webseiten, insbesondere zum Komprimieren von Webseiten, die mindestens einen statischen Block und wenigstens einen dynamischen, insbesondere durch Abruf von einem Nutzer veränderlichen Block aufweisen, die auf einem Webserver und einem Application-Server vorgehalten oder generiert werden, umfassend die Schritte:
   - Aufnehmen eines Datensatzes, umfassend mindestens einen statischen und einen dynamischen Block;
   - Feststellen jeweils einer Identität aller im Datensatz vorhandenen statischen Blöcke;
   - Feststellen, ob statische Blöcke mit derselben Identität in einem Blockspeicher schon als komprimierte Blöcke gespeichert sind und
     - entweder Ersetzen des im Datensatz vorhandenen statischen Blocks durch einen komprimierten Block dann, wenn der statische Block im Blockspeicher als komprimierter Block gespeichert ist;
     - oder Komprimieren des statischen Blocks und Abspeichern im Blockspeicher als komprimierter Block dann, wenn der statische Block noch nicht im Blockspeicher gespeichert ist und Ersetzen des statischen Blocks durch den komprimierten Block;
   - Absenden eines Sende-Datensatzes, bei welchem die statischen Blöcke durch komprimierte Blöcke ersetzt sind und der die dynamischen Blöcke enthält.

2. Verfahren nach Anspruch 1, wobei die Identität eines statischen Blocks anhand einer, aus ihm selbst gewonnenen Kennung (Fingerprint) festgestellt wird.

3. Verfahren nach einem der vorhergehenden Ansprüche, insbesondere nach Anspruch 2, wobei die aus der Identität gewonnene Kennung jedem komprimierten Block zugehörig im Blockspeicher gespeichert und beim Feststellen der Identität zum Vergleich mit den Identitäten der statischen Blöcke im Datensatz verwendet wird.

4. Verfahren nach einem der vorhergehenden Ansprüche, insbesondere nach einem der Ansprüche 2 oder 3, dadurch gekennzeichnet, daß die Kennung eine Prüfsumme über den statischen Block umfaßt.

5. Verfahren nach einem der vorhergehenden Ansprüche, insbesondere nach Anspruch 4, dadurch gekennzeichnet, daß die Kennung die Länge des statischen Blocks umfaßt.

6. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß zusätzlich zu jedem komprimierten Block der dazugehörige statische Block abgespeichert wird.

7. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß die Daten nach dem DEFLATE-Verfahren komprimiert werden.

8. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß die dynamischen Blöcke im wesentlichen unverändert, insbesondere unkomprimiert im Sende-Datensatz enthalten sind.

9. Verfahren nach einem der vorhergehenden Ansprüche, insbesondere nach einem der Ansprüche 1 bis 7, dadurch gekennzeichnet, daß mindestens ein dynamischer Block komprimiert und als komprimierter dynamischer Block in den Sende-Datensatz eingefügt wird.

10. Verfahren nach einem der vorhergehenden Ansprüche, insbesondere nach Anspruch 9, dadurch gekennzeichnet, daß abhängig von einer momentanen Arbeitsbelastung eines Servers, insbesondere des Webservers, dynamische Blöcke komprimiert oder unkomprimiert abgesendet werden.

11. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß jedem letzten Block eines Datensatzes ein Endblock zur Bezeichnung des Endes des Datensatzes hinzugefügt wird.

12. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß jeder Block des Datensatzes ein Tag zur Kennzeichnung als dynamischer oder als statischer Block insbesondere an erster Stelle hinzugefügt wird.

13. Verfahren nach einem der vorhergehenden Ansprüche, insbesondere nach Anspruch 12, dadurch gekennzeichnet, daß die Tags als HTML-Kommentar ausgebildet sind.

14. Verfahren nach einem der vorhergehenden Ansprüche, insbesondere nach einem der Ansprüche 12 oder 13, dadurch gekennzeichnet, daß die Tags von einem Parser eingefügt werden, der auf einem Application-Server läuft und auf abgelegte Templates zugreift, die er Block für Block analysiert und denen er die Tags hinzufügt.

15. Verfahren nach einem der vorhergehenden Ansprüche, insbesondere nach Anspruch 14, dadurch gekennzeichnet, daß die mit den Tags markierte unkomprimierte Webseite einem Prekompressor zugeführt wird, der entsprechend markierte Blöcke komprimiert, die zusammen mit unkomprimierten Blöcken einem Postkompressor übergeben werden.

16. Verfahren nach einem der vorhergehenden Ansprüche, insbesondere nach Anspruch 15, dadurch gekennzeichnet, daß der Postkompressor schritthaltend zur Datenbearbeitung durch den Prekompressor ein neues Dokument, vorzugsweise nach dem Gzip-Format erstellt.

17. Verfahren nach einem der vorhergehenden Ansprüche, insbesondere nach Anspruch 16, dadurch gekennzeichnet, daß der Postkompressor nach Bildung eines Gzip-Headers die vom Prekompressor übergebenen Blöcke unmittelbar mit der Übergabe in das neue Dokument einreiht.

18. Verfahren nach einem der vorhergehenden Ansprüche, insbesondere nach Anspruch 17, dadurch gekennzeichnet, daß der Postkompressor unkomprimierte Blöcke durch einen Header, vorzugsweise einen Header entsprechend dem DEFLATE-Format kennzeichnet.

19. Verfahren nach einem der vorhergehenden Ansprüche, insbesondere nach einem der Ansprüche 15 bis 18, dadurch gekennzeichnet, daß der Postkompressor einen zusätzlichen END-Block erstellt und zur Kennzeichnung eines Endes der Datenblöcke dem neuen Dokument hinzufügt.

20. Verfahren nach einem der vorhergehenden Ansprüche, insbesondere nach einem der Ansprüche 16 bis 19, da-

durch gekennzeichnet, daß der Postkompressor eine Checksumme, insbesondere eine CR-32-Prüfsumme und die Größe des unkomprimierten neuen Dokuments für einen Trailer errechnet.

21. Verfahren nach einem der vorhergehenden Ansprüche, insbesondere nach Anspruch 9, dadurch gekennzeichnet, daß dynamische Blöcke in Abhängigkeit
von mindestens einem ihnen innewohnenden Merkmal, insbesondere in Abhängigkeit von ihrer Größe vorkomprimiert gespeichert werden.

22. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß
eine Kompression von Datenblöcken nur dann vorgenommen wird, wenn diese eine voreinstellbare Mindestgröße überschreiten.

23. Datenverarbeitungseinrichtung zum Komprimieren von dynamischen Webseiten, insbesondere zum Komprimieren von dynamischen Webseiten, die mindestens einen statischen Block und mindestens einen dynamischen Block aufweisen, umfassend
einen Webserver (**10**) und einen Application-Server (**11**), gekennzeichnet durch einen Prekompressor, der so ausgebildet ist, daß bei Aufforderung durch den Webserver (**10**) an den Application-Server (**11**) eine dynamische Webseite zu generieren, der Prekompressor eine unkomprimierte Webseite in folgenden Schritten bearbeite:

- Feststellen jeweils einer Identität aller im Datensatz vorhandenen statischen Blöcke;
- Feststellen, ob statische Blöcke mit derselben Identität in einem Blockspeicher schon als komprimierte Blöcke gespeichert sind und
  - entweder Ersetzen des im Datensatz vorhandenen statischen Blocks durch einen komprimierten Block dann, wenn der statische Block im Blockspeicher als komprimierter Block gespeichert ist;
  - oder Komprimieren des statischen Blocks und Abspeichern im Blockspeicher als komprimierter Block dann, wenn der statische Block noch nicht im Blockspeicher gespeichert ist und Ersetzen des statischen Blocks durch den komprimierten Block;
- Absenden eines Sende-Datensatzes, bei welchem die statischen Blöcke durch komprimierte Blöcke ersetzt sind und der die dynamischen Blöcke enthält.

24. Datenverarbeitungseinrichtung nach Anspruch 23, dadurch gekennzeichnet, daß der Parser (**12**) so ausgebildet ist, daß die aus der Identität gewonnene Kennung jedem komprimierten Block zugehörig im Blockspeicher gespeichert und beim Feststellen der Identität zum Vergleich mit den Identitäten der statischen Blöcke im Datensatz verwendet wird.

25. Datenverarbeitungseinrichtung nach einem der Ansprüche 23 oder 24, dadurch gekennzeichnet, daß der Webserver (**10**) derart ausgebildet ist, daß abhängig von seiner momentanen Arbeitsbelastung dynamische Blöcke komprimiert oder unkomprimiert abgesendet werden.

26. Datenverarbeitungseinrichtung nach einem der Ansprüche 23 bis 25, dadurch gekennzeichnet, daß der Parser (**12**) derart ausgebildet ist, daß Tags von ihm eingefügt werden, wobei der Parser (**12**) auf dem Application-Server (**11**) läuft und auf abgelegte Templates zugreift, die er Block für Block analysiert und denen er die Tags hinzufügt.

27. Datenverarbeitungseinrichtung nach einem der Ansprüche 23 bis 26, dadurch gekennzeichnet, daß ein Prekompressor (**14**) vorgesehen ist, welchem die mit den Tags markierten unkomprimierten Webseiten zugeführt werden und der zu Komprimierung entsprechend markierter Blöcke ausgebildet ist, und daß ein Postkompressor (**15**) vorgesehen ist, dem die komprimierten Blocke zusammen mit unkomprimierten Blocke vom Prekompressor (**14**) übergeben werden.

28. Datenverarbeitungseinrichtung nach einem der Ansprüche 23 bis 27, dadurch gekennzeichnet, daß der Postkompressor (**15**) schritthaltend zur Datenverarbeitung durch den Prekompressor (**14**) und zur Erstellung eines neuen Dokuments vorzugsweise nach dem Gzip-Format ausgebildet ist.

29. Datenverarbeitungseinrichtung nach einem der Ansprüche 23 bis 28, dadurch gekennzeichnet, daß der Postkompressor (**15**) zur Bildung eines Gzip-Headers und zur unmittelbaren Einreihung der vom Prekompressor (**14**) übergebenen Blöcke mit der Übergabe in das neue Dokument ausgebildet ist.

30. Datenverarbeitungseinrichtung nach einem der Ansprüche 23 bis 29, dadurch gekennzeichnet, daß der Postkompressor (**15**) zur Kennzeichnung unkomprimierter Blöcke durch einen Header, vorzugsweise einen Header entsprechend dem DEFLATE-Format ausgebildet ist.

31. Datenverarbeitungseinrichtung nach einem der Ansprüche 23 bis 30, dadurch gekennzeichnet, daß der Postkompressor (**15**) zur Erstellung eines zusätzlichen END-Blocks und zur Hinzufügung des END-Blocks zur Kennzeichnung eines Endes der Datenblöcke zum neuen Dokument ausgebildet ist.

32. Datenverarbeitungseinrichtung nach einem der Ansprüche 23 bis 31, dadurch gekennzeichnet, daß der Postkompressor (**15**) zur Bildung einer Checksumme und Herleitung der Größe des unkomprimierten neuen Dokuments und zur Erstellung eines Trailers ausgebildet ist.

33. Datenverarbeitungseinrichtung nach einem der Ansprüche 23 bis 32, dadurch gekennzeichnet, daß der Prekompressor derart ausgebildet ist, daß dynamische Blöcke in Abhängigkeit von mindestens einem ihnen innewohnenden Merkmal, insbesondere in Abhängigkeit von ihrer Größe vorkomprimiert gespeichert werden.

34. Datenverarbeitungseinrichtung nach einem der Ansprüche 23 bis 33, dadurch gekennzeichnet, daß der Prekompressor derart ausgebildet ist, daß eine Kompression von Datenblöcken nur dann vorgenommen wird, wenn diese eine voreinstellbare Mindestgröße überschreiten.

Hierzu 5 Seite(n) Zeichnungen

Nummer: DE 101 46 356 A1
Int. Cl.[7]: H03M 7/30
Offenlegungstag: 24. April 2003

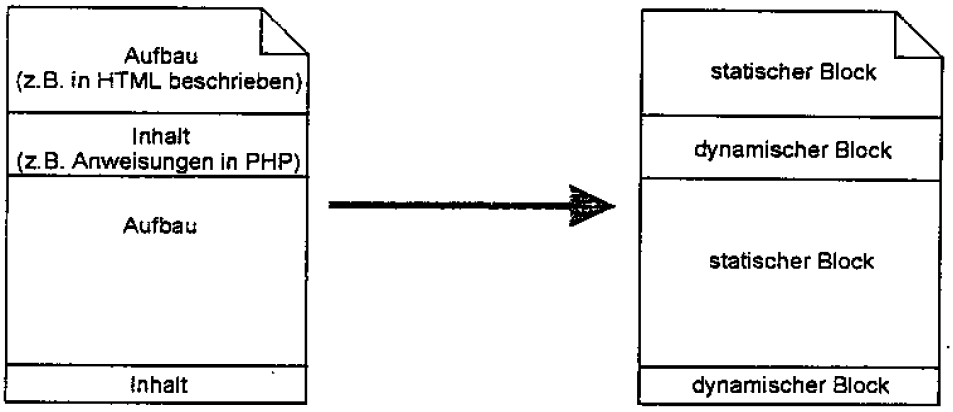

Fig. 1

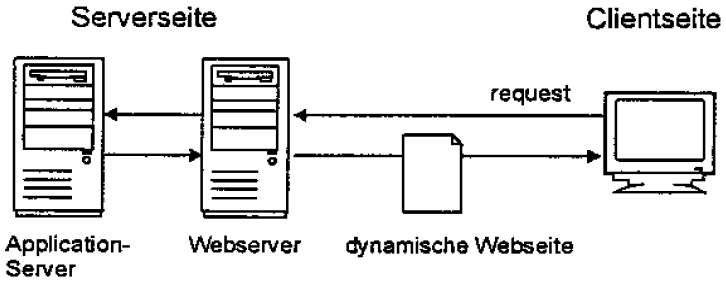

Fig. 2

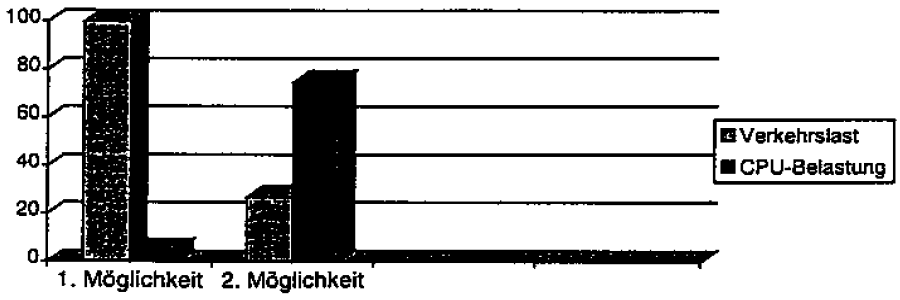

Fig. 3

Nummer: DE 101 46 356 A1
Int. Cl.[7]: H03M 7/30
Offenlegungstag: 24. April 2003

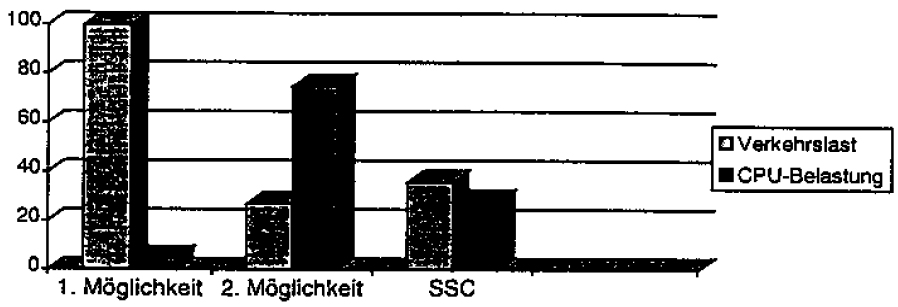

Fig. 4

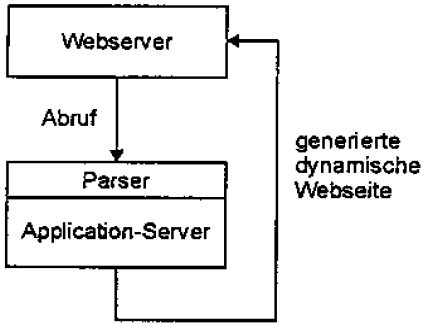

Fig. 5

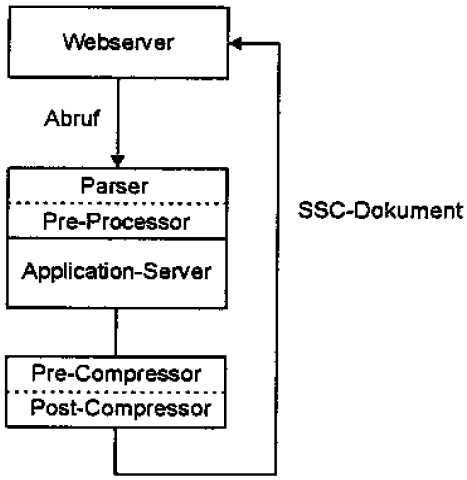

Fig. 6

Nummer: DE 101 46 356 A1
Int. Cl.[7]: H03M 7/30
Offenlegungstag: 24. April 2003

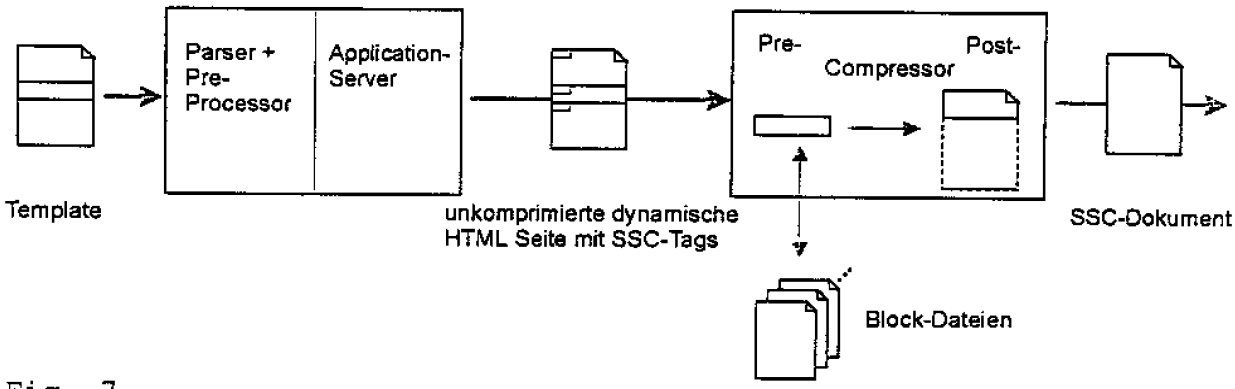

Fig. 7

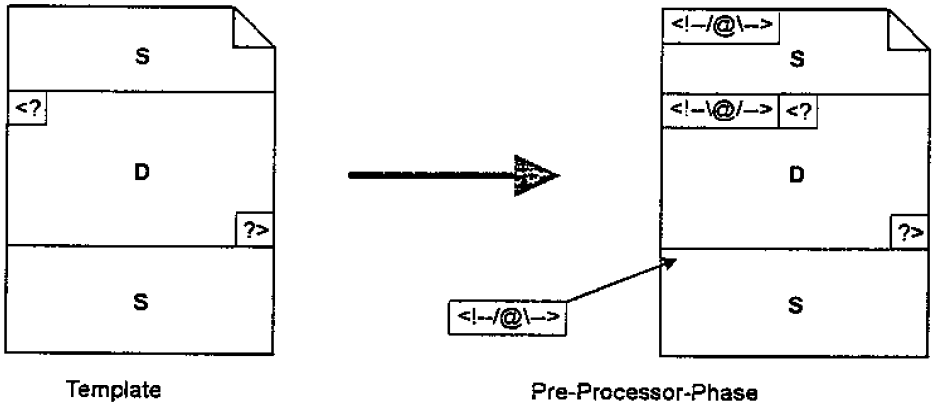

Fig. 8

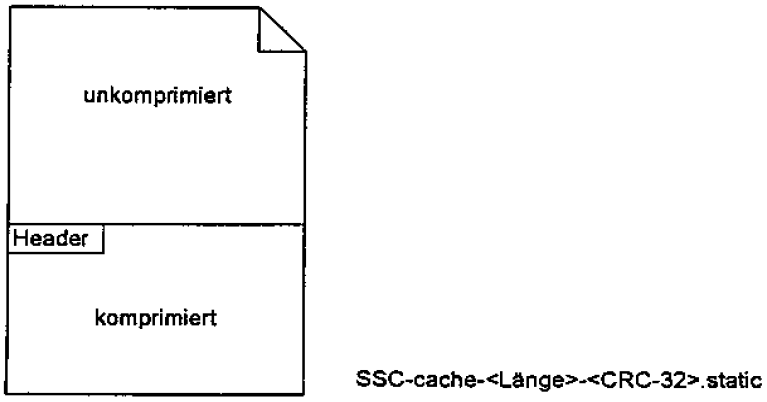

Fig. 9

Nummer: DE 101 46 356 A1
Int. Cl.[7]: H03M 7/30
Offenlegungstag: 24. April 2003

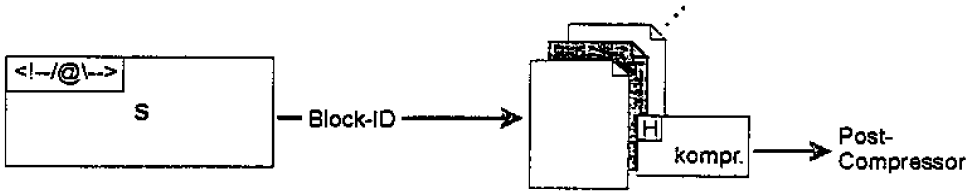

Fig. 10

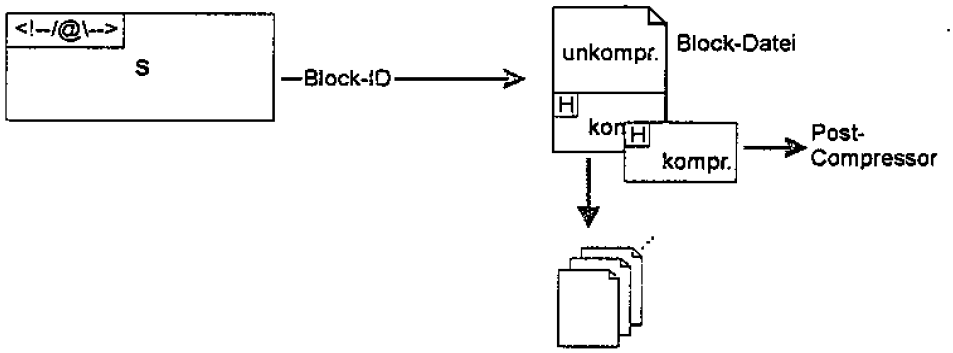

Fig. 11

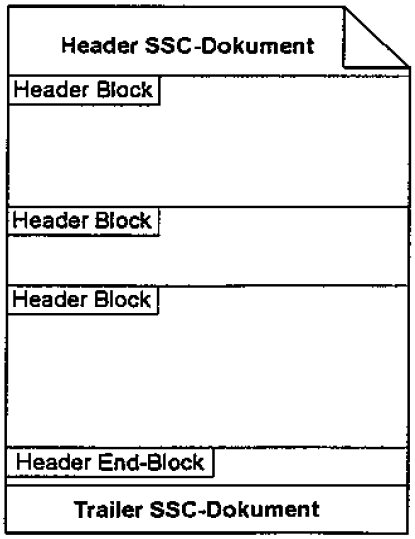

Fig. 12

Nummer: DE 101 46 356 A1
Int. Cl.[7]: H 03 M 7/30
Offenlegungstag: 24. April 2003

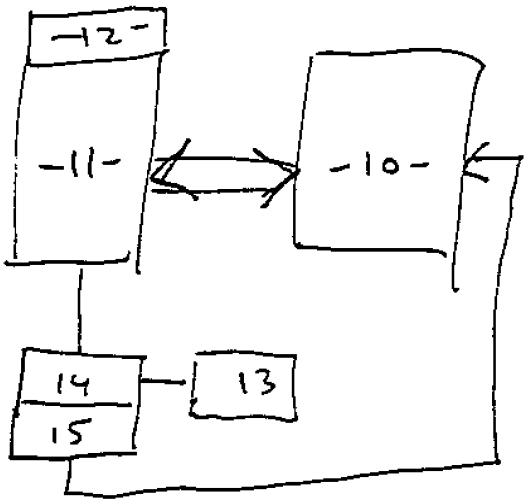

Fig. 13